Arduino
Calculateur capacité
Batterie
Lithium-Ion 2 éléments

Table des matières

Projet 1 _ Minimal

Le but est de réaliser une mesure rapide de la capacité réelle d'une batterie Li-ion deux éléments du type couramment utilisé dans les APN.

Le test est réalisé sous charge constante et continue, avec un arrêt de la charge lors de la détection du seuil de tension basse de la batterie.

Electronique

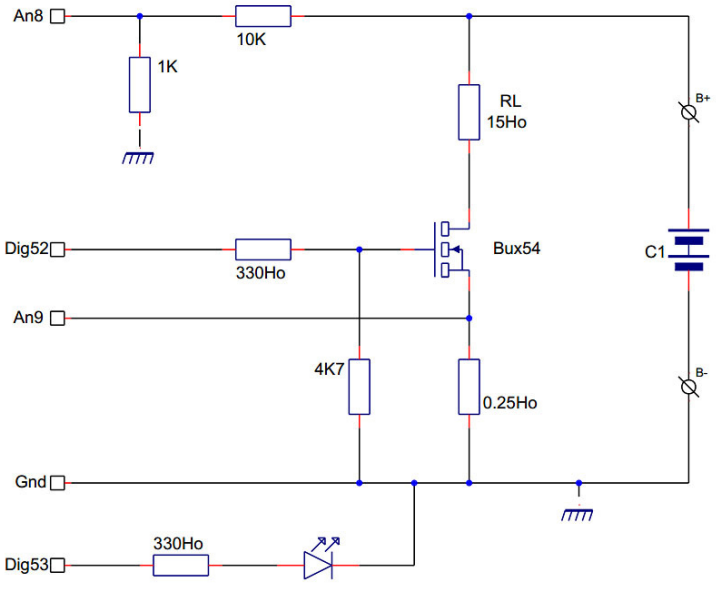
Utilisation d'un Arduino Mega et d'un shield LCD standard, avec l'utilisation des ports :

- Digitaux : Un port permettant de commuter la charge, un second relié a une Led permettant d'étalonner l'horloge interne.
- Analogiques : Un port pour les états des touches du shield, deux pour les mesures de tension et d'intensité.

L'utilisation de la tension de référence interne de 2.56v pourrait être remplacée avantageusement par une source externe calibrée (TL1431 par ex). Cette valeur de 2.5v est un compromis entre les faibles tensions à mesurer et le fonctionnement des touches du shield LCD travaillant sur une base 5v.

La mesure de tension est réalisée via un diviseur 10, celle de l'intensité via un shunt 0.25Ho ce qui nous fait des tensions à mesurer de l'ordre de 150mv pour une charge de 15Ho. Attention a l'emplacement de la masse de référence.

La charge RL est constituée par une résistance de puissance montée sur radiateur de valeur comprise entre 8 et 25 Ho.



Logiciel

Ecrit sans optimisation : Le cycle est mis en route par la touche Haut, mis en pause par la touche Bas et réinitialisé par la touche droite, un flag se chargeant de la mémorisation et de la gestion de l'état. L'arrêt est provoqué par la détection d'un niveau de tension basse.

Les 4 états de fonctionnement sont :

- S(top) : Effacement des mesures en attente nouveau cycle
- R(un) : Décharge en cours
- P(ause) : Décharge en pause
- E(nd) : Batterie déchargée

L'étalonnage des mesures est réalisé via 3 constantes, deux pour la mesure de tension et d'intensité, la troisième pour la compensation de la chute de tension due au câblage en charge.

Les mesures sont réalisées toutes les secondes ce qui permet une intégration des valeurs mesurées en mAh. Pour permettre une réactivité des touches correctes le cycle logiciel n'utilise pas directement une temporisation de (1s - t cycle) mais un cycle de 10ms la mesure n'étant effectuée que tout les 100 cycles environs, cette valeur de division étant utilisée comme variable d'ajustement.

```
#include <LiquidCrystal.h>

//Définition paramètres shield clavier - LCD
LiquidCrystal Lcd(8,9,4,5,6,7);

const int PortAnKey = 0;
int AnalogKey = 0;
int ValKey = 0;
int MemoKey = 20;
const byte KeyHaut =0x9;
const byte KeyBas =0x13;
const byte KeyDroit =0x1;
const byte KeyGauche =0x1E;
const byte KeySel =0x20; // Non utilisable avec ref 2.5v
const int KeyNone =0x20;

// Entrées Sorties

const byte PortDisch = 52;
const byte PortTest = 53;
const byte PortAnI = 9;
const byte PortAnU = 8;
const float CalI = 10.2;  // Constante intensité en fonction R shunt
const float CalUI = 0.001;  // Constante rattrapage tension shunt + câblage
const float CalU = 0.02746;  // Constante U en fonction diviseur

// Divers
int Cpt1 = 0;       // Compeur cycle 1s
int CptS = 0;       // Compteur cycle 10s envoi trame série
char FlagRun ='S';   // Mode de fonctionnement en cours
word TimeRun =0;     // Temps en s de décharge
word ValIRaw = 0;    // Valeurs lecture analog
word ValURaw = 0;
float ValI = 0;      // Valeur tension en mV
```

```
float ValU = 0;      // Valeur intensité en mA
float Capa = 0;      // Valeur Capacité en mAs
float ValUstop = 6;  // Tension d'arret en V
byte tmpTest=0;

void setup() {   // Initialisation ######################################################

  Lcd.begin (16,2);
  Lcd.clear();

  analogReference(INTERNAL2V56);
  //analogReference(EXTERNAL);

  pinMode (PortDisch, OUTPUT);
  digitalWrite (PortDisch,LOW);
  pinMode (PortTest, OUTPUT);
  digitalWrite (PortTest,LOW);

  Serial.begin(115200);

}

void loop() {      //Boucle principale ################################################


    AnalogKey = analogRead(PortAnKey); //Gestion des Touches ++++++++++++++++++++++++++
    ValKey = (AnalogKey / 32) + 1;

    if (MemoKey != ValKey){
       MemoKey = ValKey;

       if (ValKey == KeySel) {}

       if (ValKey == KeyDroit) {
         if (FlagRun !='R') {FlagRun='S';}}

       if (ValKey == KeyGauche) {}

       if (ValKey == KeyHaut) {
         FlagRun='R';}

       if (ValKey == KeyBas) {
         FlagRun='P';}

    }

  Cpt1 ++;     // Temporisation pour affichage 1 x secondes et calcul capacité.+++++++++
  if (Cpt1 > 95) {
    Cpt1=0;

    LectureAnalog();
    AfficheLcd ();

    if (FlagRun=='S') {                 // RAZ mesure capacité ++++++++++++++++++++++
      Capa=0;
      TimeRun=0;
      CptS=0;
      digitalWrite (PortDisch, 0);
    }
    else if (FlagRun !='E') {
```

```
    if (ValU < ValUstop) {              // Arrêt décharge tension basse ++++++++++++
      FlagRun='E';
      EnvoiSerie();
    }
    if (FlagRun =='R') {
      digitalWrite (PortDisch, 1);       // Décharge ++++++++++++++++++++++++++++++++
      Capa += ValI;
      tmpTest =! tmpTest;
      digitalWrite (PortTest, tmpTest);

      if (CptS ==0) {EnvoiSerie();}      // Envoi port com 1x minutes +++++++++++++++
      CptS ++;
      if (CptS == 60) {CptS=0;}

      TimeRun ++;
    }
    else {
      digitalWrite (PortDisch, 0);       // Arrêt décharge ++++++++++++++++++++++++++
    }
   }
  }

  delay (10);

}



// Lecture et affichage U - I ========================================================

void LectureAnalog() {


  ValIRaw = analogRead(PortAnI) ;
  ValURaw = (analogRead(PortAnU)) ;
  ValU = (ValURaw * CalU) + (ValIRaw * CalUI);
  ValI = ValIRaw * CalI;
}

void AfficheLcd () {

  Lcd.clear();
  Lcd.setCursor(1,0);
  Lcd.print ("U=");
  Lcd.print ((ValU));
  Lcd.setCursor(9,0);
  Lcd.print ("I=");
  Lcd.print (int(ValI));
  Lcd.setCursor(0,1);
  Lcd.print(FlagRun);
  Lcd.print(":");
  Lcd.print(TimeRun);
  Lcd.setCursor (8,1);
  Lcd.print ("C=");
  Lcd.print (Capa/3600);
}

// Envoi données port com toutes les 60 secondes de décharge ==========================
void EnvoiSerie () {
```

```
    Serial.print (TimeRun);
    Serial.print ("  ;  ");
    Serial.print(ValURaw);
    Serial.print (" ; ");
    Serial.print(ValIRaw);
    Serial.print (" ; ");
    Serial.print(ValU);
    Serial.print (" ; ");
    Serial.print(ValI);
    Serial.print (" ;     ");
    Serial.println (Capa);

  }
```

Révisions document

v1.00	14/03/2014	Première diffusion
v1.01	13/10/2015	Corrections schema